Attorney's Docket No.: 07072-152001 / EMC-

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DYNAMIC DEMONSTRATION OF UNIMPLEMENTED ALGORITHMS

APPLICANT: EITAN BACHMAT, HAGIT BACHMAT AND RON ARNAN

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL486016417US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

2-26-02
Date of Deposit

[illegible]
Signature

Leroy Jenkins
Typed or Printed Name of Person Signing Certificate

DYNAMIC DEMONSTRATION OF UNIMPLEMENTED ALGORITHMS

FIELD OF INVENTION

This invention relates to data-storage systems, and in particular, to monitoring the performance of algorithms executed by such systems.

BACKGROUND

The success of a data-storage system depends, to a great extent, on its ability to reliably read and write large amounts of data very quickly. To do so, the data-storage system is often called upon to perform several tasks, each of which can be performed using any one of several different algorithms.

Configuring a data-storage system typically includes selecting, for each task that the data-storage system is called upon to perform, a particular algorithm for performing that task. In some cases, the algorithm is selected, in part, on the basis of the manner in which the data-storage system is expected to be used. In other cases, the algorithm is simply a default algorithm.

Once the data-storage system has been configured, it becomes difficult to confirm that the algorithm selected for a particular task is truly the best choice for that task. Although it is possible to determine how well the chosen algorithm has performed over a specified interval, this is not enough. It is always possible that another algorithm, had it been executing during that same interval, would have done better.

One approach to comparing the performance of two algorithms is to allow the first algorithm to operate during a first interval and then to allow the second algorithm to operate during a second interval. There are several flaws in this approach. First, the usage patterns of the data-storage system, upon which the performance depends, may differ during the first and second intervals. Second, this approach rapidly becomes more time-consuming as more algorithms are compared. Third, it is inconvenient to have to reconfigure the data-storage system simply to test other algorithms.

Because of the difficulty associated with comparing the performances of different algorithms, many data storage systems are needlessly crippled by algorithms that are inappropriate for their usage patterns.

SUMMARY

The invention is based on the recognition that a data-storage system can statistically characterize the manner in which it is used. The data-storage system can then use the resulting characterization to simulate the performance of algorithms that are not actually being executed on the data-storage system. The simulated performances of these algorithms can then be directly compared with the performance of an incumbent algorithm that is actually being executed. This results in a data-storage system that can dynamically demonstrate how different algorithms would have performed had they been operating on the same input data stream as that being operated upon by the incumbent algorithm.

The invention achieves the foregoing result by simulating the execution of competing algorithms for the same task and continuously monitoring the performance of each competing algorithm. The performance of one or more of the competing algorithms is then compared with a corresponding performance of whichever algorithm is the incumbent algorithm.

In some practices of the invention, a method for providing data indicative of the performance of a competing algorithm and an incumbent algorithm includes evaluating an incumbent-algorithm score indicative of a performance of an incumbent algorithm. The performance of a competing algorithm executing in place of the incumbent algorithm is then simulated. On the basis of the simulation, a competing-algorithm score predictive of a performance of the competing algorithm is evaluated. The competing-algorithm score and the incumbent-algorithm score are then made available, typically to an output device.

In other practices of the invention, providing data includes monitoring the incumbent-algorithm score and the competing-algorithm score during a selected interval. Examples of such data include a ratio indicative of an extent to which the competing-algorithm score exceeds the incumbent algorithm score during the selected interval.

In other practices of the invention, simulating performance includes obtaining meta-data characterizing an input-data stream being provided to the incumbent algorithm. Such meta-data can include, but is not limited to, statistics descriptive of the input data-stream during the selected interval. The performance of the competing algorithm is then simulated using this meta-data. The result of this simulation is predictive of the performance of the competing algorithm were it to operate on the input-data stream characterized by the meta-data.

Other practices of the invention include providing data indicative of a performance of a competing algorithm and an incumbent algorithm in a data-storage system. Methods for doing so include the statistical characterization of a usage pattern of the data-storage system; and on the basis of the statistical characterization, the simulation of a performance of the competing algorithm were it to execute on the data-storage system in place of the incumbent algorithm. In certain embodiments, statistical characterization of a usage pattern of the data-storage system includes generating meta-data that characterizes an input data-stream to the data-storage system.

Additional practices of the invention include evaluating actual performance of the incumbent algorithm in response to the usage pattern and simulating performance of the competing algorithm in response to the usage pattern. The resulting data, which is indicative of a comparison between the actual performance of the incumbent algorithm and the simulated performance of the competing algorithm, is then provided to an output device.

Another practice of the invention includes comparing how well each of a plurality of algorithms performs a task. This practice includes simulating execution of a competing algorithm operating on the input stream; evaluating, on the basis of the simulation, a competing-algorithm performance of that competing algorithm; and evaluating the actual performance of an incumbent-algorithm operating on the input stream. The resulting data, which is indicative of a comparison between the incumbent algorithm's actual performance and the competing algorithm's simulated performance, is then made available.

The invention also includes systems for carrying out the foregoing methods. One such system includes a data-condenser configured to receive a data-stream and to generate meta-data characterizing the data stream. The data-condenser communicates with a competing-algorithm simulator that generates data indicative of a performance attribute of a competing algorithm when the competing algorithm operates on a data-stream characterized by the meta-data. A tournament manager receives the performance attributes and makes available output data indicative of a comparison between a performance attribute of the competing algorithm and a corresponding performance attribute of an incumbent algorithm.

These and other features of the invention will be apparent from the following detailed description and the figures, in which:

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 shows a data-storage system; and

FIG. 2 shows an architecture embodying the invention.

DETAILED DESCRIPTION

Referring to FIG. 1, a data-storage system **10** includes an I/O subsystem **12** that provides an interface between one or more hosts **14a-c** and one or more mass-storage elements **16a-d**. The hosts **14a-c** request the I/O subsystem **12** to perform particular tasks. In a data-storage system **10**, these tasks are most typically requests to store data and requests to retrieve data. The mass-storage elements **16a-d** are most often disk drives. However, the principles of the invention described herein are independent of the particular media used to store data.

A service processor **18** in communication with the I/O subsystem **12** controls the operation of the I/O subsystem **12**. The service processor **18** can be accessed either at the site of the data-storage system **10** or remotely, through a data communication link.

A variety of algorithms exist for performing the tasks requested by the hosts **14a-c**. Using the service processor **18**, a system operator can control which of these algorithms is used to perform a given task. However, the best algorithm for performing a task depends on the usage patterns of the data-storage system **10**. These usage patterns can change with time.

For example, in one algorithm for retrieving data, the I/O subsystem **12**, in response to a host's request for data, pre-fetches additional, related data. To the extent that a request for particular data is followed shortly thereafter by a request for the related data, this strategy reduces latency. However, if there is no correlation between a request for particular data and subsequent requests for related data, this strategy consumes system resources unnecessarily, thereby increasing latency.

Another source of latency arises because the hosts **14a-c** might issue requests faster than the I/O subsystem **12** can process those requests. The I/O subsystem **12** manages these competing demands by maintaining queues and distributing requests among those queues. There are several well-known algorithms available for managing queues, each of which has its own set of advantages and disadvantages. As just one example, the I/O subsystem **12** may choose to process brief tasks before processing more time-consuming tasks. The advantage of this algorithm is that users requesting simple operations will, consistent with their expectations, experience lower latencies. The disadvantage of this algorithm is that if there are too many brief tasks, time-consuming tasks may be excessively delayed.

In both of the foregoing examples, the best algorithm for accomplishing a task depends on the manner in which the I/O subsystem **12** is being used at around the time the I/O subsystem **12** receives a request to perform that task. In many cases, this is unpredictable.

The selection of the best algorithm for performing a task is thus an empirical process in which the performances of candidate algorithms are compared with each other as those algorithms operate on a live input stream. This might be achieved by executing a first algorithm for a first interval followed by executing a second algorithm for a second interval. The flaw in this approach, however, is that the live input stream is a non-stationary random process. As a result, statistics characterizing a live input stream during that first interval may not be the same as those characterizing the live input stream during the second interval.

FIG. 2 shows an architecture for enabling a system operator to more readily compare the performances of different algorithms performing the same task on what is effectively the same input stream. The constituents of the architecture shown in FIG. 2 represent processes being executed by one or more processors. These processes can be executed by one or more processors in the I/O subsystem **12**, by the service processor **18** in communication with the I/O subsystem **12**, or by a combination of one or more processors in the I/O subsystem **12** and the service processor **18**.

The architecture shown in FIG. 2 simulates the performance of one or more competing algorithms on the same live data stream. In the illustrated system, an incumbent-algorithm process **20** receives data in an input data-stream **22** and processes that data in the conventional manner. At one time, the incumbent algorithm may have been the best algorithm for performing a particular task. However, this is no guarantee that the incumbent algorithm will forever be the best algorithm. As the nature of the input data-stream **22** changes over time, the best algorithm for processing that input data-stream **22** might also change.

Although FIG. 2 shows an input data-stream **22** entering the incumbent-algorithm process **20**, this need not be the case. The input data-stream **22** can be any data-stream that can be used to compare the performance of an incumbent algorithm with the performance of one or more competing algorithms.

An incumbent-algorithm performance-monitor **24** in communication with the incumbent-algorithm process **20** continuously monitors the performance of the incumbent algorithm. The performance attribute measured by the incumbent-algorithm performance-monitor **24** depends, in part, on the nature of the task being performed by the incumbent algorithm. Generally, the performance attribute measures, directly or indirectly, the latency that results from application of the incumbent algorithm to the input data-stream **22**.

The choice of a performance attribute depends in part on the task to be performed. For example, if the task is to manage a cache memory, a suitable performance attribute might be a hit ratio that indicates the probability that data sought is already in the cache memory. Other performance attributes might include response time, bandwidth, and throughput. Whichever performance attribute is selected, the output of the incumbent-algorithm performance-monitor **24** is an incumbent-algorithm score **26** indicative of the performance of the incumbent algorithm.

The input data-stream **22** provided to the incumbent-algorithm process **20** is also provided to a data-condenser **28**. The data-condenser **28** extracts meta-data from the input data-stream **22** and passes that meta-data to one or more competing-algorithm simulators **30a-c**. Meta-data refers to data that characterizes the input data-stream **22**. For example, meta-data might refer to the number of requests for data stored within a particular range of locations. In effect, meta-data is data that is used to statistically characterize the input data-stream **22**. The particular type of meta-data extracted by the data-condenser **28** depends on the nature of the task being performed.

Each of the competing-algorithm simulators **30a-c** simulates the application of a corresponding competing algorithm for performing the same task that is being performed by the incumbent algorithm process **20**. The results of the simulations are then provided to one or more competing-algorithm performance-monitors **32a-c**. These competing-algorithm performance-monitors **32a-c** measure the same performance attribute measured by the incumbent-algorithm performance-monitor **24**. Each competing-algorithm performance-monitor **32a-c** generates competing-algorithm scores **34a-c** indicative of their respective algorithms' performances.

The competing-algorithm scores **34a-c** and the incumbent-algorithm score **26** are all provided to a tournament manager **36**. The tournament manager **36** monitors these scores over time and periodically provides output data **38** to an output device **40**. This output data **38** includes data indicative of a comparison between the performance of the incumbent algorithm and the performances of one or more of the competing algorithms. A system operator viewing this output data **38** on the output device **40** is then in a better position to determine whether to replace the incumbent algorithm by one of the competing algorithms.

The nature of the output data **38** that the tournament manager **36** provides to the output device **40** is programmable. The output data can include data indicative of an extent to which the competing-algorithm score **32a-c** exceeds the incumbent-algorithm score **26** and how consistently the competing-algorithm score **32a-c** exceeds the incumbent-algorithm score **26**. In one practice of the invention, the tournament manager **36** can handicap the competing-algorithm scores **32a-c** by amounts that reflect the system resources consumed in replacing the incumbent algorithm with the competing algorithm.

A system as illustrated in FIG. 2 thus enables a data-storage system **10** to demonstrate the relative performances of competing algorithms operating in parallel on statistically identical input streams. It does so by continuously estimating, for each competing algorithm, what the performance of that competing algorithm would have been had that competing algorithm instead been the incumbent algorithm. The system then provides output data indicative of whether the competing algorithm would have been better than the incumbent algorithm over a statistically significant interval. This output data enables a system operator to assess the desirability of replacing the incumbent algorithm with a competing algorithm.

Having described the invention, and a preferred embodiment thereof, what we claim as new and secured by letters patent is: